A Novel Group Coordination Protocol for Collaborative Multimedia Systems

H.-P. Dommel and J. J. Garcia-Luna-Aceves
{peter | jj}@cse.ucsc.edu
Computer Communication Research Group
Baskin School of Engineering
University of California
Santa Cruz, CA 95064, USA

ABSTRACT

Group collaboration in distributed multimedia environments extends gradually to larger groups and wide-area networks. While reliable multicasting has made significant advancements in recent years, effective mechanisms to synchronize and coordinate work within large multicast groups and across long distances are still lacking. Group coordination is here understood as the mediated access to shared remote resources in synchronous groupwork, as for example in telecollaboration and distributed simulation environments, complementing protocols for group membership, media synchronization and reliable ordered multicast. A comparative analytic model for known classes of group coordination mechanisms, ranging from socially mediated control to floor control in ring and tree topologies, is presented. It is shown that hierarchical group coordination is the most efficient and scalable approach to date. Based on these findings, a novel protocol is described, which dynamically organizes participants in a multi-level control tree and aggregates resource sharing directives on the paths between interacting stations.

1 INTRODUCTION

Increasing deployment of IP-multicast [6] has brought a new generation of collaborative multimedia applications to mainstream computing. While previous collaboration tools were proprietary, monolithic, and designed for small-scale collaboration in local area networks, newer applications are geared towards larger sessions with wide geographic range. For example, distributed interactive simulations and distance learning sessions can easily involve hundreds of participants. Although reliable multicasting and multicast routing technology have advanced considerably, efficient group coordination support for applications characterized by synchronous and wide-area groupwork is still lacking.

Our goal is to extend support for dynamic group coordination to wide-area networks and large multicast groups, characterized by intermittent connectivity and heterogeneous multimedia information. The central idea behind group coordination protocols is to establish a resource-sharing discipline, counter end-application misbehavior, and incorporate end-user demand for a specific Quality-of-Service, allowing to throttle bandwidth utilization and impacting informed resource allocation. It is still a subject of controversy, whether group coordination services are to be tailored to the application-level, or whether they should be offered as a middleware component, which a variety of applications demanding for resource cooperation can tap into. To this date, a variety of group coordination protocols have been proposed under different names, but have not been specified in more detail, compared, or placed in a consistent methodological framework.

We regard coordination of group activities on shared resources as a distributed middleware service tailored toward the semantics of the media involved, rather than specific applications. We focus in this paper on a particular form of group coordination, known as floor control. By being granted a "floor", a user attains a permission for exclusive usage of a resource. Floor allocation establishes clear rules for turn-taking, and prevents race conditions, indefinite resource holding, and unfairness in resource access patterns. Participating stations agree on a specific floor policy, concerning service order and priorities. The spectrum of control can range from lenient to strict, reflecting characteristics of tasks and interaction styles, such as user roles, usage quotas, or resource contention periods. As a component within a general coordination architecture for many-to-many groupwork, floor control coexists with protocols for reliable ordered multicast and media synchronization at a sub-application level. Orchestration of multiparty groupwork with fine-grained and fair floor control is an open research problem. In this paper, we offer a fresh look on this topic by making the case for hierarchical floor control.

The remainder of the paper is organized as follows: Section 2 discusses cornerstones of related work. Section 3 presents a brief overview and comparison of existing classes of group coordination protocols. We find that floor control over a shared propagation tree, corresponding to the underlying end-to-end reliable multicast tree, represents the most scalable and efficient way to store and forward control information. In particular, it allows to exploit the hierarchical nature of multicast groups, supports selective control packet dissemination to multicast subgroups, and distributes load about floor state keeping across the multicast tree, without compromising ease of implementation. Section 4 describes the operation of the Hierarchical Group Coordination Protocol (HGCP), a novel approach for floor control in shared propagation trees. Section 5 concludes the paper.

2 RELATED WORK

Coordination problems such as initiation conflicts and resource competition in multimedia conferencing have been reported in [9]. Early efforts on coordination of group activities led to the conception of floor control for networked databases and telecollaboration systems [20]. Various precursors of collaborative applications [2, 5, 21, 23] contained some form of proprietary floor control for small sessions with broadcast of control information and centralized coordination. The centralized method is comparable to a sender-initiated multicast solution, which severely limits the capacity of a group coordination system. First, one station must maintain and process a large amount of state information associated with each resource and floor contender. Second, the sender as the floor controller must process many floor requests in a short time and may corrupt the entire session, if failing. An alternate mechanism has been proposed in the teleconferencing architecture by Aguilar *et al.* [1], in which a distributed, task-activated floor control for teleconferencing is used, as a high-level analogy to collision-sensing in channel access. In Yavatkar and Lakshman's approach [25], floor control is understood as a transport level service, realized as a token-based concurrency control scheme for media flows, which has been implemented on top of the reliable multicast protocol XTP. Lately, floor control has appeared in a variety of commercial and experimental systems. For example, the MBone application suite has been enriched by a moderated bulletin board [15], which allows users in a larger scaled conference to post questions and receive attention from specific participants. Amir *et al.* [3] suggest to intertwine floor control with a rate-adaptation mechanism, which throttles media stream transmission according to the interest or capabilities of a heterogeneous receiver set. Only sources being granted the floor are allowed to consume bandwidth, and floor control directives can be used to reserve future bandwidth shares.

Report Documentation Page

1. REPORT DATE **1998**	2. REPORT TYPE	3. DATES COVERED **00-00-1998 to 00-00-1998**

4. TITLE AND SUBTITLE **A Novel Group Coordination Protocol for Collaborative Multimedia Systems**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**6**	

3 GROUP COORDINATION PARADIGMS

We propose a basic taxonomy which divides known group coordination algorithms into two classes: *implicit group coordination* (IGC) and *explicit group coordination* (EGC) algorithms.

Taxonomy

IGC algorithms mark the state of resource usage with assertions on local variables determined through one or more rounds of messages or probing of remote resources' states. Since no token entity is explicitly exchanged, such algorithms are also referred to as "permission-based". The challenge for this class lies in the globally consistent state keeping of local assertions. IGC schemes are inherently contention-based, since sites must actively compete for a floor. Disadvantages of IGC are the need for continuous sensing and the failure-prone observation of remote resource states, either by man or machine, with a strong likelihood of collisions due to network latency, or lack of coordination. As a consequence, contending stations, being unaware of each others' immediate actions, may experience collisions by trying to access a resource at the same time.

In contrast, EGC algorithms explicitly exchange a unique token among sites. Obtaining this token symbolizes the right to access a resource, thereby dispersing floor contention by regulated passing of permission place-holders. Hence, this class is also referred to as "token-based". In addition to the token being passed among sites, control messages are exchanged to request, deny, reserve, or grant the token, depending on the control scheme used. A shortcoming of EGC schemes is the higher cost of token tracking, in order to ensure token uniqueness and authenticity. EGC schemes generally operate on a defined infrastructure, which logically organizes the receiver set.

Both classes operate based on the dichotomy between a *floor holder* (FH), which designates the current legitimate user of a resource, and a *floor coordinator* (FC) for that resource, which determines the legitimate floor holder at a given time. Often, the FH is also the FC. We assume that a generic group coordination protocol uses the following basis set of control messages: a station sends a FRQ (floor request) to the FC to demand access to a resource; the FC either replies with FGT (floor grant) to grant permission to access the resource (when the floor is free), or FDY (floor deny) to signal that the floor is taken or reserved; a station transmits a FRL (floor release) message to relinquish the floor.
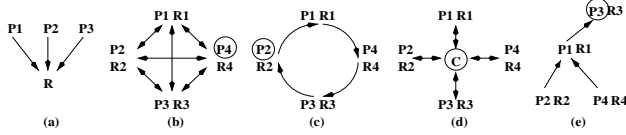

Figure 1: Group coordination geometries.

The crucial difference between algorithms lies in the assignment of these roles to various nodes during a session, which impacts the routing of control messages. Figure 1 depicts typical coordination scenarios, where a circled entity marks the FC or FH: (a) several participants P_i try to gain control of a centralized resource R with limited or no coordination; (b) P_i communicate directly with each other in order to obtain a consensus about floor holdership; (c) a ring structure linearizes interaction and control passing between pairs of nodes; (d) one centralized station serves as FC in a star-topology, which is a one-level subcase of (e); (e) control messages are propagated along branches of a multi-level tree topology towards the current FH.

The goal for all algorithms is the correct and efficient determination of the FC and FH per resource at every turn in a collaborative session. The taxonomy in Figure 2 corresponds to the constellations in Figure 1.

Comparative Analysis

Our analytic comparison of known classes of group coordination protocols is a first attempt to characterize the efficacy of interactive behavior of people and processes from a resource contention
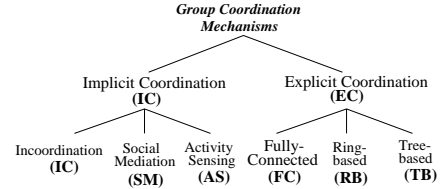

Figure 2: Group coordination mechanisms.

perspective. The goal is to assess how much overhead is intrinsic to the various protocols with regard to control state management, based on topology and signaling. We take into consideration an average floor request arrival rate, task length, and network propagation delay. In our model, a collaboration session $C_s = (S, L)$ in a computer network consists of a set of stations S (sites, nodes) and a set of links $L \subset S \times S$. Each station, in an intranetwork or internetwork, hosts a session member (user, process), serves remote stations with local resources, and is client for remote resources. Links are either reliable (no loss, but possibly delayed delivery), resilient (timely delivery, but likely some loss), or without service guarantees.

For reasons of tractability, we make the following assumptions: the underlying dissemination model is broadcast, i.e., a host sends each message only once to the network interface, where it is IP-multicast to the receivers; message delivery between hosts is FIFO, and no station failures occur; user interface and host processing overhead are negligible for each station; the interarrival rate of floor requests is Poisson, given that there is no indication for cross-correlations between subsequent floor requests; floor allocation is greedy, that is, the first registered request is fulfilled, and others are discarded and may be resubmitted later; finally, the floor holding time to execute a task is on the average the same and normalized to unity.

β_1	average "think" time before floor token arrival
β_2	average "think" time at floor token presence
γ	average processing time for a floor directive
δ	duration of average activity period
$n\epsilon$	processing and unicasting overhead to n receivers
η	efficacy of a floor control protocol
G	average offered floor request load
ι	average duration of idle time
λ	floor request interarrival rate
m	average number of stations in session
n	average number of active stations in session
ν	average vulnerability period
τ	average propagation delay

Table 1: Analysis parameters.

Table 1 summarizes the notation. ν is the time, during which a station's attempt to access a resource can be intercepted by another station. We assume for all protocols that τ signifies the average propagation delay for multiple routing hops between stations coalesced into one hop. A packet must hence traverse on the average the same number of hosts on the path from the sender to a group of receivers. $G = \delta \times \lambda$ is the offered request load on floors, including new and previously denied, and resubmitted floor requests. For IGC schemes, γ denotes the average contention period, whereas for EGC schemes, γ is the average processing time per control packet.
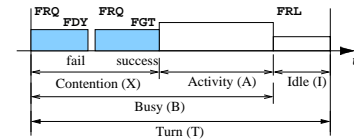

Figure 3: Conceptual turn-taking model.

A turn-taking model, shown in Figure 3, serves as the conceptual foundation for our analysis. A turn consists of three stages, a

contention period X, an activity period A, and an idle time I. A timeline for a turn shows how long it takes on the average for an individual station to acquire a specific floor. The various schemes allocate these periods differently to grant and revoke floors on resources. Based on this model, we define the *efficacy* η of a group coordination protocol with multicast support as the ratio of floor usage time vs. overall turn length, given by Eq. 1:

$$\eta = \frac{\bar{U}}{\bar{T}} = \frac{\bar{U}}{\bar{X} + \bar{A} + \bar{I}} \qquad (1)$$

Incoordination (ICIC) refers to unawareness about other stations' activities, caused by minimal user interface feedback or high-latency networks, and may lead to completely unsynchronized resource access. This is the case for early collaboration systems such as [20] and long-distance conferencing with unpredictable delays.

Theorem 1 *The efficacy of ICIC is*

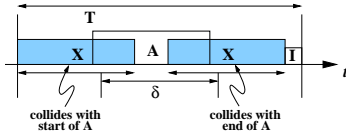$$\eta_{ICIC} = \lambda \delta e^{-2\lambda\delta} \qquad (2)$$

Figure 4: ICIC timeline.

Proof: Figure 4 shows a prototypical turn under incoordination. The proof is the same as for medium access in an ALOHA channel [4]. Messages can intercept each other during contention, hence the vulnerability interval is twice the task length. Therefore, because request arrivals are Poisson, the probability that a task is successful is $e^{-2\lambda\delta}$. The success probability times the number of arrivals in one activity period results in Eq. 2. □

Social Mediation (ICSM) leaves floor negotiation to social conformities among session members using available awareness cues. If there is remote activity, a user contending for the floor withdraws for a random time and reclaims the floor, when the remote activity subsides. This is the case for modern POTS conferencing systems. Both ICIC and ICSM incur no implementation cost with regard to system-aided coordination, however, reliability is low and likelihood of conflict is high. It is possible that a single host and user become moderator and floor controller for all other hosts, creating a bottleneck system-wise and user-wise for reasons mentioned above, if many receivers demand control decisions from this one host at the same time.

Theorem 2 *The efficacy of ICSM is*

$$\eta_{ICSM} = \frac{\delta}{\delta + e^{2\lambda\gamma'}\left[\frac{e^{\lambda\gamma'}-1-\lambda\gamma'}{\lambda\gamma'(1-e^{-\lambda\gamma'})} + \gamma' + \tau + \iota + \frac{1}{\lambda}\right]} \qquad (3)$$
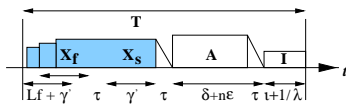

Figure 5: ICSM timeline.

Proof: A prototypical timeline of ICSM is depicted in Figure 5. The success probability is denoted as P_s, the failure probability is $P_f = 1 - P_s$. The average utilization period lasts $\bar{U} = \delta P_s$, and the length of the average busy period $\bar{B} = \bar{X} + \bar{A}$ is determined by the time needed to handle unsuccessful floor requests in the failed contention period X_f and successful requests in X_s, with $\bar{B} = (1 - P_s)X_f + P_s X_s$. γ' is the time needed to sense and react to resource states. An average failed turn attempt consists of a geometrically-distributed indefinite number

(of L) of interarrival times of floor requests with duration $\bar{f}$ sec (average time between failed floor-request arrivals), plus the duration of any given floor request (γ'). The values for $\bar{L}$ and $\bar{f}$ have been derived in [22]. Substituting our notation in these results, we obtain $\bar{L} = e^{\lambda\tau}$ and $\bar{f} = (\lambda\tau)^{-1} - e^{-\lambda\tau}/(1 - e^{-\lambda\tau})$, respectively. Accordingly, the average time of a failed turn attempt equals $X_f = \left[\frac{e^{\lambda\gamma'}-1-\lambda\gamma'}{\lambda\gamma'(1-e^{-\lambda\gamma'})}\right] + \gamma' + \tau$. P_s equals the probability that no activity packet arrives in a vulnerability period ν of $2\gamma'$ sec., i.e., $P_s = P[0\ packets\ in\ \nu] = e^{-2\lambda\tau}$. A successful turn is $X_s = \gamma' + \delta + \tau$. Finally, the expected idle time is $\bar{I} = \iota + \frac{1}{\lambda}$. Substituting into Eq. 1, we obtain Eq. 3. □

Activity Sensing (ICAS) is a special class of floor control proposed in [8], where shared activities on resources are monitored by a background process at the session layer, in order to sense which site currently operates on the resource. This mechanism is comparable to collision sensing in medium access control. If an ICAS process at a local site detects remote activity on a resource, it denies the local user the floor, until remote activity subsides. In that way, a distributed collective of activity sensing agents ensures better tracking of resource states, and helps to prevent conflicts. The disadvantage of this scheme is its high implementation cost and its reliance on short link latencies, which makes it usable only in LANs.

Theorem 3 *The efficacy of ICAS is*

$$\eta_{ICAS} = \frac{\delta}{\delta + \tau + \frac{1}{\lambda} + e^{\lambda\tau}(\gamma + 2\tau + \iota)} \qquad (4)$$
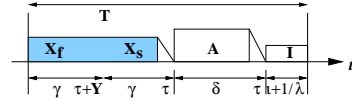

Figure 6: ICAS timeline.

Proof: The timeline is shown in Figure 6. The average utilization period is $\bar{U} = \delta P_s$, with $P_s = P[0\ packets\ in\ \nu] = e^{-\lambda\tau}$. The average length of a successful busy period is simply $\delta + \gamma + 2\tau$. The length of an average unsuccessful activity period consists of one truncated activity lasting γ sec, followed by one or more similarly truncated activities sent within time Y sec, where $0 \leq Y \leq \tau$. The expected value of Y is $\bar{Y} = \tau - \frac{1}{\lambda}(1 - e^{-\lambda\tau})$ [22]. The length of the average busy period is then $\bar{B} = \gamma + 2\tau - \frac{1}{\lambda} + e^{-\lambda\tau}(\delta + \tau + \frac{1}{\lambda})$. The average idle interval is again $\bar{I} = \iota + \frac{1}{\lambda}$. Substitution into Eq. 1 yields Eq. 4. □

The class of explicit group coordination comprises three protocol families, as well. Common to all families is the exchange of a floor token symbolizing exclusive access to the shared resource. It must be assured that token transmission is reliable and that all sites execute the same protocol correctly to avoid conflicts based on duplicated, lost or forged tokens.

In *Fully-connected Group Coordination* (ECFC), each station is directly connected to every other station. In the simplest case, a station acquires the floor by sending a FRQ to the other $n - 1$ stations, which send a FGT message back if they do not hold the floor. If a station is FH for this floor, it sends a FDY message, and follows up with a FGT as soon as it releases the floor. If several stations ask for the same floor from each other, control packet sequence numbers (or another election mechanism) determine the next FH. The complexity of ECFC grows quadratically with the number of hosts. For each floor transaction, $2(n - 1)$ messages are required, following the Ricart-Agrawala mutual exclusion solution [19]. Newer quorum or tree-based [18] algorithms for obtaining an exclusive token can reduce the number of messages to $O(log n)$, where n is the session size. However, these schemes do

not scale well, since the network can easily be flooded with too many control messages, and significant delay variations between hosts can eventually deadlock collaborative turn-taking.

Theorem 4 *The efficacy of ECFC is*

$$\eta_{ECFC} = \frac{\delta}{\delta + 3(\gamma + \tau) + (n - 1)\epsilon + \iota + \frac{1}{\lambda}} \qquad (5)$$
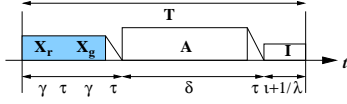
Figure 7: ECFC timeline.

Proof: The timeline for ECFC is shown in Figure 7. The time per station to send, receive, and update floor information with every other station is $3(\gamma + \tau)$. An extra processing time ϵ from $n - 1$ stations must be added. Substitution into Eq. 1 results in Eq. 5. □

In *Ring-based Coordination* (ECRB), a token rotates in a well-defined sequence among sites and holding the token determines the current floor holder, with two subcases: (a) a token claim can be decided upon before arrival, or during the arrival hold-time, or only while it visits the local station, and (b) a token can be transferred immediately after a forward from a station to the next one in the ring, or only after the releasing station received an acknowledgment. ECRB is particularly suited to ensure totally ordered and atomic delivery, but does not scale well, since the token walk time is proportional to the receiver set. In addition, the predefined traversal order may cause artificial delays in the interactions between particular stations. A token-ring based collaboration system has been evaluated by Pendergast [17].

Theorem 5 *The efficacy of ECRB is given by*

$$\eta_{ECRB} = \frac{\delta(1 - e^{-\lambda(\beta_1+\beta_2)})}{\frac{n}{2}(\tau + \gamma + \beta_2) + \delta(1 - e^{-\lambda(\beta_1+\beta_2)}) + \iota + \frac{1}{\lambda}} \qquad (6)$$
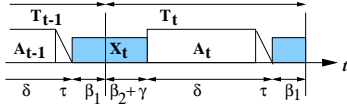
Figure 8: ECRB timeline.

Proof: The timeline for ECRB is depicted in Figure 8. The average utilization is again $\bar{U} = \delta P_c$, with P_c as the probability that the token is available in the period $X = \beta_1 + \beta_2$. This period includes the time γ needed to transfer the token to the neighbor station. The token cycle time is a function of the session size. On the average, a floor token has to cycle through half the ring to be transferred. The probability that a floor can be claimed by a user is $P_c = 1 - e^{-\lambda(\beta_1+\beta_2)}$. Accordingly, the average turn lasts $\bar{T} = \frac{n}{2}(\tau + \gamma + \beta_2) + \bar{U}$. The idle time is again $\iota + \frac{1}{\lambda}$. Substituting $\bar{T}$, $\bar{U}$ and $\bar{I}$ into Eq. 1 yields Eq. 6. □

Tree-based Coordination (ECTB) algorithms conduct floor management in a logical tree structure. Such a tree may reflect the actual multicast routing tree, or mirror the end-to-end reliable multicast tree. Control messages are passed along branches of the tree in a parent-child relation, which reflects multicast group membership. No specific site alone is hence burdened with the obligation to make floor allocation decisions, and tokens can wander freely across the tree branches, without being cast into a specific traversal order other than what multicast group membership expresses. While various tree-based reliable multicast protocols have been proposed for highly scalable conferencing [12, 14, 24], no group coordination mechanism or applications have been developed yet. The goal for such a protocol is to keep the tree depth

minimal, in order to keep the number of hops for control transmissions low. A special case of ECTB is a star-topology, which is a tree of depth one with a root and $n - 1$ descendants. The moderator-driven question board [15] exemplifies this scheme. Such a centralized scheme suffers as a communication bottleneck primarily from overload when the request rate increases. In our analysis, focused on multicast support for turn-taking, we did not account for this additional individual host processing cost.

Theorem 6 *The efficacy of ECTB is*

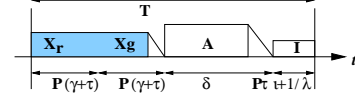$$\eta_{ECTB} = \frac{\delta}{\delta + 2\gamma + 3\tau + \iota + \frac{1}{\lambda}} \qquad (7)$$

Figure 9: ECTB timeline.

Proof: The timeline for ECTB is depicted in Figure 9. With multicasting, a request-reply pair takes two γ and τ, plus another τ to signal completion of the turn. A close correlation between the control tree and the end-to-end multicast tree is assumed. The idle period is again $\iota + \frac{1}{\lambda}$. Substituting into Eq. 1 gives Eq. 7. Note that we normalize the pathlength for a generic tree coordination protocol to $\bar{P} = 1$, assuming that the host tree mirrors the multicast routing tree. Under this assumption, each of the discussed protocols must on the average traverse the same number of hosts on the path from the sender to a group of receivers. □

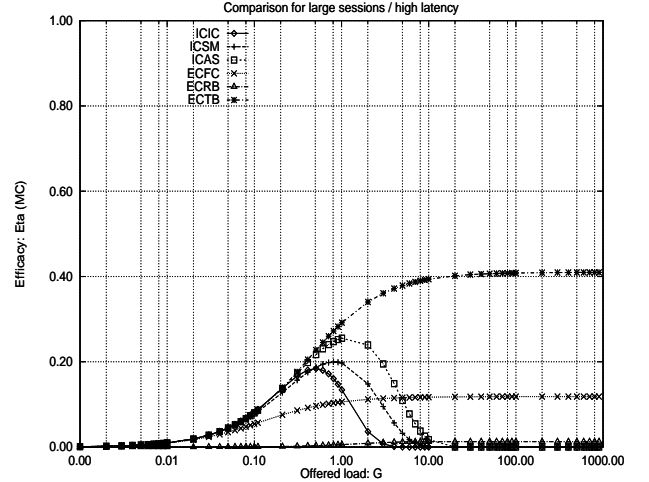Figure 10: Efficacy for large groups and high link latency.

Results

Traces from MBone sessions [13], sampled over a period of several hundred hours, indicate a group size ranging from 150 to over 550 participants. As a snapshot to indicate the advantages of tree-based group coordination, Figure 10 plots the efficacy of discussed schemes for a large session, $n = 300$, and wide-area collaboration, $\tau = 400ms$. Such a scenario can occur for Internet collaboration, with an assumed latency that has been observed to be the minimum for acceptable audio conferencing quality [9].

We can see that a generic tree-based floor-control protocol exhibits superior stability and scalability with regard to higher loads. With an aggregated control scheme, floor control remains effective despite a large number of directives being exchanged among participants. Since such a protocol taps into the given infrastructure of a pre-built host tree, there is no additional setup cost. A more comprehensive analysis and thorough discussion of other scenarios, varying group size and latency, can be found in Ref. [7].

4 PROTOCOL DESCRIPTION

We outline the basic operation of the Hierarchical Group Coordination Protocol (HGCP). For a more detailed description and proof of correctness for HGCP we refer to [7]. Our goal is to exploit the best-effort delivery mechanism of IP multicast in a scalable and efficient tree protocol for reliable group coordination. The protocol consists of two stages: (1) control tree construction, and (2) control message dissemination. In order to consistently account for floor control messaging, we assume ordered message delivery between hosts. Ordering in multicast trees has been tackled for example by Jia [10].

We distinguish between four types of nodes in a propagation tree: the source node is the current FH and transmits information to the receiver set; hop nodes are positioned on the path from the source to the receivers, and are called extra nodes, if they are not contained in the receiver set; destination nodes are members of the receiver set. Each of these nodes can contend for any floor at any time and become FH. In a multicast collaboration session with multiple floors and floor holders, it is not practical to manage separate control trees, one per floor. Similar to the concurrent reliable multicast scheme proposed in [12], we hence assume a single shared control tree with a branching factor B, which can be re-hung as a control tree with any other node as the root and FH, while preserving the property that all nodes have B children. Ideally, the control tree correlates closely to the multicast routing tree, as built by multicast routing protocols such as DVMRP, CBT, or PIM [16].

We focus on an approach comparable to receiver-initiated multicast [12], which lets contending stations in the group-coordinated session maintain state information about a particular floor. Floor state information can hence be passed along between contending stations, without requiring the controller station to contact each station. For this purpose, each station maintains two timers. Timer one is employed to detect whether a floor control message has been lost. Timer two is used to delay repetitive control messaging to the controller, in the hope that a near-by station is able to report an update from the controller station. The collective state information about all floors is hence distributed across the control tree. Note that HGCP is independent of a particular reliable multicast implementation, but relies on the failure-free transmission of control packets rendered by such a protocol. Transmission of control packets concerning floor states is based on the abstraction of a *propagation group*, which consists of processes (users, hosts) scattered in the network, which are interested in coordinating their activities on a particular resource set. Propagation groups can be built by tapping into the information provided by a session directory service.

The control tree of HGCP organizes group participants into a hierarchy of subgroups or *coteries*. Each such coterie has a *group manager*, which acts as a representative for all other members in the group. The group manager is responsible for querying control states of floors for its group members. For reasons of simplicity and more efficient group coordination, it can be assumed that FC and FH are identical for a particular floor at all times. If multiple FH are allowed in parallel for a particular resource, it is necessary to separate the FC and FH roles.

A construction method for a control tree for group coordination on top of a multicast routing tree is outlined in [12]. A node x in the resulting hierarchical acknowledgment tree, whose edges are directed from children to their parents, is labeled with a unique label $l(x)$, which is the prefix of all children of x. Labeling is top-down, and the maximal length of labels in a tree reflects its depth. The label alphabet can consist of any set of symbols with a defined order (in the simplest case integers). Note that labels in the tree remain constant for the lifetime of the session, except in cases when nodes fail or new nodes join a subtree. Rare cases when relabeling is necessary are discussed in [12]. Adding a node in the tree involves only the new node as a child and its parent, while deletions require relabeling of the subtree of the deleted node. The labeling scheme allows for *implicit routing* of control messages, nodes can remain anonymous in the interaction, yet address each other efficiently by using label prefix comparison,

and only a single tree is needed for concurrent multicast. Support for implicit labeling and routing at the multicast router level has been proposed by Levine and Garcia-Luna in [11]. A sample scenario is shown in Figure 11. In this scenario, a ternary control tree is labeled from alphabet $\{0, 1, 2\}$, with $FH = d$ located at $l(d) = 110$.
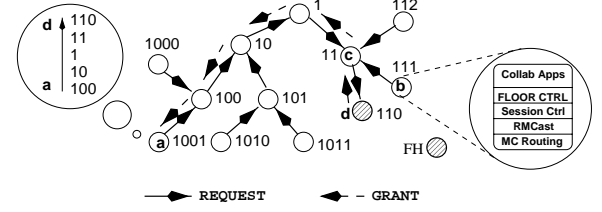

Figure 11: Sample HGCP scenario.

An example, how control relegation works, is the following: at all times, nodes maintain a local record about the labeled position of the FH in the tree. Any node can assume this role, when being granted the floor. A control directive (CD) is a tuple of the format

$$(\{l_s\}, \{l_d\}, \texttt{t}, \texttt{TTL}, \texttt{ID}, \texttt{f})$$

containing an ordered list of source labels $\{l_s\}$, a list of destination labels $\{l_d\}$, a sequence number or timestamp t, a time-to-live TTL, an identifier ID, and a floor f. The floor field f allows to discern media types, priorities, and the operational semantics (e.g., access control) of a resource. This format allows to send and receive composite floor directives and reduce the amount of control traffic, which is particularly critical for large and highly-interactive sessions with many floors and resources.

Assume that node a with $l(a) = 1001$ wants to submit a FRQ CD, and finds $FH = d$, with $l(FH) = 110$, in its local record. The request percolates up in the tree across the root node and down on the right branch according to the label prefix semantics. The label format allows for self-routing of control directives and addressing of specific nodes in the multicast group, simply by knowing their logical address, and without nodes having to identify themselves otherwise. Until FH receives a FRQ, it multicasts resource state updates to its immediate neighbors, which forward the information to the session remainder. CDs are *aggregated* at hop or extra nodes, by coalescing requests or responses from neighbor nodes into single CDs and forwarding them accordingly. Concurrent submissions are resolved by attaching a timestamp to each control packet, which gives preference to the earliest submitted request. Node d, after completion of its resource operation, multicasts back a FGT message to node a, which also informs all other nodes in the session via flooding about the new position of $FH = a$. Each node maintains an entry for each floor on the relative path direction (next hop) in the tree, where FHs are currently to be found. An FH also may keep a record that indicates the number and type of floors granted to each receiver, and establish a fairness policy based on service priorities. Furthermore, denied requests may be queued at FH and held pending, until they can be fulfilled, or are timed out. Implicit routing allows also to address coterie members in a floor-controlled multicast group by listing individual labels as destinations, which is also an elegant solution to the anycasting problem. No updates regarding the position of the FC and FH are disseminated, if these role assignments are static.

On the average, leaf nodes must compare more bits in the control routing procedure, than nodes close to the root. The label cardinality depends on the tree branching factor and the session size. A tree with n session participants and branching factor B has $log_B n$ levels, with $log_2 n$ bits needed for node labels. The virtual re-hanging procedure on the tree may cause imbalances with regard to the average pathlength and create unfairness in the time-based allocation of floors. However, since spatio-temporal information regarding control message origin is available to the respective FH, floor allocation can include also criteria such as

distance and promote uniform treatment of stations. A basic algorithm for labeling and routing of floor control messages, including only `FRQ` and `FRL` cases and separation of FC and FH, is shown in Figure 12.

```
set session graph S ← C_S;
set floor f_i ⊆ F ← state(resource r_j);
        /* initialize floors, i natural, for each resource */

protocol HGCP ();
start session C_S;
LABEL_CONTROL_TREE (S);
while C_S is running
begin
        foreach f_i do
                ROUTE_CONTROL_MESSAGE ();
end

LABEL_CONTROL_TREE (graph G)
begin
        construct tree T ⊆ E from G with root r;
        set l(r) = 1;
        LABEL_SUBTREE(r, T, l(r));
end

LABEL_SUBTREE (node s, tree T, integer β)
begin
        set α = 0;
        foreach child c of s do
                set l(c) = α ∘ β;
                increment α;
                LABEL_SUBTREE(c, T, l(c));
end

ROUTE_CONTROL_MESSAGE (tree T, node x, node c, directive p)
begin
        if |l(x)| > |l(c)|
        or if |l(x)| ≠ prefix of |l(c)|
                then route p to parent of x;
                else ROUTE_CONTROL_MESSAGE
                        ( subtree(T), children(x), node c, directive p);
                /* multicast to the children of x with matching prefix */
                HANDLE_FLOOR_DIRECTIVE( node x, node c, directive p);
                /* if local node is destination, handle floor directive */
end

HANDLE_FLOOR_DIRECTIVE (node x, node c, directive p)
begin /* for floor f */
        case p
                FRL: if c == FC
                        then set FH = nil;
                FRQ: if c == FC and FH == nil
                        then c sends FGT to x;
                        else if c == FH
                        or if FH ≠ nil
                        then FH sends FDY to coterie;
                        /* percolate floor busy message back along tree */
        end case
end
```

Figure 12: Basic HGCP Specification

5 CONCLUSION

Software to support multiparty interactivity and coordination becomes more important with improvements in networking technology, allowing more sophisticated ways of telepresence and real-time collaboration. The current state-of-the-art of reliable multicast uses single shared acknowledgment trees to warrant scalable and failure free message dissemination within large multicast groups. A primary application area for such protocols is telecollaboration, however, protocols to coordinate large groups with regard to resource access are lacking. This paper compared known solutions for permission- and token-based floor control and described the first tree-based floor control protocol embedded with reliable hierarchical multicast. An implementation and further performance assessment are under way.

6 ACKNOWLEDGMENTS

7 REFERENCES

[1] L. Aguilar, J. J. Garcia-Luna-Aceves, D. Moran, E.J. Craighill, and R. Brungardt. Architecture for a multimedia teleconferencing system. In *Proc. ACM SIGCOMM*, pages 126–136, Aug. 1986.

[2] S. R. Ahuja and J. R. Ensor. Coordination and control of multimedia conferencing. *IEEE Comm. Mag.*, 30(5):38–43, May 1992.

[3] E. Amir, S. McCanne, and R. Katz. Receiver-driven bandwidth adaptation for light-weight sessions. In *Proc. ACM Multimedia*, Seattle, WA, Nov. 1997.

[4] D. Bertsekas and R. Gallager. *Data networks*. Prentice Hall, Englewood Cliffs, N.J., 2nd edition, 1992.

[5] T. Crowley, P. Milazzo, E. Baker, H. Forsdick, and R. Tomlinson. MMConf: An infrastructure for building shared multimedia applications. In *Proc. ACM CSCW*, pages 637–650, Los Angeles, CA, Oct. 1990.

[6] S. Deering. Host extensions for IP multicasting. RFC-1112, August 1989.

[7] H.-P. Dommel and J. J. Garcia-Luna-Aceves. Efficacy of floor control protocols in distributed multimedia collaboration. *Cluster Computing*, submitted for publication 1999.

[8] J. J. Garcia-Luna-Aceves, E. Craighill, and R. Lang. Floor management and control for multimedia conferencing. In *Proc. IEEE Multimedia, 2nd COMSOC Int. Multim. Comm. Worksh.*, Ottawa, Can., Apr. 1989.

[9] E. A. Isaacs and J. C. Tang. What video can and cannot do for collaboration: a case study. *Multimedia Systems J.*, 2(2):63–73, Aug. 1994.

[10] X. Jia. A total ordering multicast protocol using propagation trees. *IEEE Trans. Par. and Dist. Sys.*, 6(6):617–627, June 1995.

[11] B. N. Levine and J. J. Garcia-Luna-Aceves. Improving internet multicast with routing labels. In *Proc. IEEE ICNP*, pages 241–250, Atlanta, GA, Oct. 1997.

[12] B. N. Levine, D. Lavo, and J. J. Garcia-Luna-Aceves. The case for concurrent reliable multicasting using shared ack trees. In *Proc. ACM Multimedia*, Boston, MA, Nov. 1996.

[13] J. Liebeherr and B. S. Sethi. A scalable control topology for multicast communication. In *Proc. IEEE Infocom*, San Francisco, Mar./Apr. 1998.

[14] J. C. Lin and S. Paul. RMTP: A reliable multicast transport protocol. In *Proc. IEEE Infocom*, pages 1414–1425, Mar. 1996.

[15] R. Malpani and L. Rowe. Floor control for large MBone seminars. In *Proc. ACM Multimedia*, pages 155–163, Seattle, WA, Nov. 1997.

[16] T. Maufer and C. Semeria. Introduction to IP multicast routing. Internet Draft, URL http://www.ietf.org/internet-drafts/draft-ietf-mboned-intro-multicast-03.txt, July 1997.

[17] M. O. Pendergast. Multicast channels for collaborative applications: design and performance evaluation. *Computer Communication Review*, 23(2):25–38, April 1993.

[18] K. Raymond. A tree-based algorithm for distributed mutual exclusion. *ACM Trans. on Comp. Sys.*, 7(1):61–77, Feb. 1989.

[19] G. Ricart and A. K. Agrawala. An optimal algorithm for mutual exclusion in computer networks. *Comm. ACM*, 24(1), 1981.

[20] S. Sarin and I. Greif. Computer-based real-time conferences. *IEEE Computer*, 18(10):33–45, Oct. 1985.

[21] K. Srinivas, R. Reddy, et al. MONET: A multimedia system for conferencing and application sharing in distributed systems. Technical report, Concurrent Engineering Research Center, West Virginia University, Morgantown, WV, Feb. 1992. CERC-TN-RN-91-009.

[22] H. Takagi and L. Kleinrock. Output processes in contention packet broadcasting systems. *IEEE Trans. Commun.*, COM 33(11):1191–1199, 1985.

[23] K. Watabe, S. Sakata, K. Maeno, H. Fukuoka, and K. Marbara. Distributed multiparty desktop conferencing system: MERMAID. In *Proc. ACM CSCW*, pages 27–38, Los Angeles, CA, Oct. 1990.

[24] R. Yavatkar, J. Griffioen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *Proc. ACM Multimedia*, pages 333–344, San Francisco, Nov. 1995.

[25] R. Yavatkar and K. Lakshman. Communication support for distributed collaborative applications. *Multimedia Systems J.*, 2(2):74–88, Aug. 1994.